Abstract

In the current scenario, with the transpiring big data explosion, data sets are often too large to fit completely inside the computers’ internal memory. In efficient processes, speed is not an option, it is a must. Hence every alternative is explored to further enhance performance, by expanding in-place memory storage that enables more data to be resident in the memory, eliminating operation latency, and even deploying an in-memory database (IMDB) system where all the data can be kept in memory. However, the technique of in-memory data handling is still at an infant stage and not viable in the current scenario. To tackle this problem a hierarchical hashing scheme is discussed where only one component of a big data structure resides in the memory. In this paper two data structures are explored: 1) Map which is implemented as self-balancing binary search trees or more commonly Red Black Trees and 2) Unordered Map which is based on hashing with chaining technique. Serialization and deserialization operations are also performed to free the internal memory and preserve the data structure object for later use. Operations such as read, write are performed, along with documentation of the results and illustrations of visual representations of the two algorithmic
data structures.

References

1. Jeffrey Scott Vitter, Purdue University "Dealing with massive data"
2. G. DeCandia, D. Hastorun, M. Jampani et al., Dynamo: Amazon's highly available
   key-value store, OSR, 2007.
3. Hao Zhang, Gang Chen, Kian-Lee Tan et al., "In-Memory Big Data Management and
   Processing: A Survey"
5. J. Ousterhout, P. Agrawal, D. Erickson et al., The case for ramclouds: Scalable
6. F. Li, B. C. Ooi, M. T. Ozsu, and S. Wu, Distributed data management using
8. Standard C++ Library reference:
   http://www.cplusplus.com/reference/unorderedmap/unordered_map/ Cpp Reference
   Documentation
    Reference Documentation
12. D. J. DeWitt, R. H. Katz, F. Olken et al., Implementation techniques for main memory
    database systems, in SIGMOD 84, 1984.
13. R. B. Hagmann, A crash recovery scheme for a memory-resident database system, TC,
    1986.
14. T. J. Lehman and M. J. Carey, A recovery algorithm for a high performance
    memory-resident database system, in SIGMOD 87, 1987.
15. M. H. Eich, Mars: The design of a main memory database machine, in Database
    Springer Berlin Heidelberg, 2013
18. S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan, Distributed online aggregations, in PVLDB 09,
    2009.
19. T. J. Lehman and M. J. Carey, A study of index structures for main memory database
    management systems, in PVLDB 86, 1986.
20. J. Rao and K. A. Ross, Cache conscious indexing for decision-support in main memory,
    platforms, in ICDE 13, 2013.
22. T. Brown, F. Ellen, and E. Ruppert, A general technique for non-blocking trees, in

**Index Terms**

Computer Science Programming Language

**Keywords**

Map, Unordered Map, Boost C++, Serialization, Hierarchical hashing, Memory management